

Logic-analyzer interface assists in 68030 program debugging

Real-time program developers can use all the tools they can get. Once you learn how to build a trace-interface circuit for a logic analyzer, you can gain external access to the MC68030's debugging feature and filter out unimportant bus cycles as well.

Don Atkins, *Motorola Inc*

Aids for debugging a program during the development stage are undeniably indispensable. Debugging a real-time program is particularly vexing because you can't stop the program while you evaluate the execution of events. In real-time robotic, automotive, and industrial applications, the target system has to keep operating to determine if the control program in question is functioning properly.

To facilitate program development, Motorola's family of MC68000 μ P's include an internal facility that allows you to trace instructions while you debug a program. The MC68030 μ P provides additional output pins that give you external access to the internal facility during real-time operation. By decoding the μ P's extra outputs and control signals, you can monitor real-time program operation on a logic analyzer.

External tracing lets you examine key parameters

while the μ P is executing a program. For example, you can detect an instruction boundary, which indicates the end of an instruction cycle during program execution; you can detect if the execution unit is discarding prefetched operations in the pipeline, and you can detect whether read and write accesses are hitting the on-chip cache memories.

To maximize the potential of external tracing, you must fulfill two requirements. First, the external-tracing circuitry must be synchronized to the μ P's bus activity. Second, the circuitry must generate a sampling signal that qualifies conditions and eliminates extraneous clock cycles that you don't want or need to analyze.

Tracing takes advantage of μ P features

Before you can build a trace-interface circuit, you must understand how the μ P performs some of its basic operations. The MC68030 has two modes for transferring data. In the synchronous mode, it has to wait until the device it wants to communicate with sends the $\overline{\text{STERM}}$ (Synchronous Termination) signal, indicating to the μ P that the device is ready for data transfer. In the asynchronous mode, the 68030 has to wait till the device responds with the $\overline{\text{DSACK1}}$ or $\overline{\text{DSACK0}}$ signal (or both). To trace data into and out of the μ P, both modes must generate a sampling signal when valid data is on the bus.

During program execution, the 68030 prefetches instructions and operands to keep three stages of its

By decoding the μ P's control signals and extra outputs, you can monitor the real-time operation of a program on a logic analyzer.

instruction pipeline full. The pipeline allows the execution unit to perform concurrent operations of as many as three words of a single instruction or as many as three consecutive instructions. Instructions predominantly execute sequentially; however, it is possible that

TABLE 1—EFFECTS OF STATUS SIGNAL ASSERTION

IF STATUS IS ASSERTED FOR	THEN
1 CLOCK PERIOD	SEQUENCER IS AT AN INSTRUCTION BOUNDARY AND WILL BEGIN EXECUTION OF NEXT INSTRUCTION
2 CLOCK PERIODS	SEQUENCER IS AT AN INSTRUCTION BOUNDARY BUT WILL NOT BEGIN THE NEXT INSTRUCTION IMMEDIATELY BECAUSE OF EITHER A PENDING TRACE EXCEPTION OR A PENDING INTERRUPT EXCEPTION
3 CLOCK PERIODS	AN MMU ADDRESS-TRANSLATION-CACHE MISS HAS OCCURRED; PROCESSOR WILL BEGIN TABLE SEARCH OR EXCEPTION PROCESSING WILL BEGIN FOR THE FOLLOWING: RESET, BUS ERROR, ADDRESS ERROR, SPURIOUS INTERRUPT, AUTOVECTORED INTERRUPT, OR F-LINE INSTRUCTION (NO COPROCESSOR RESPONDED)
4 OR MORE CLOCK PERIODS	PROCESSOR IS HALTED

the execution unit won't operate on some prefetched data. For example, when a program encounters a change-in-flow instruction, such as a branch, jump, subroutine-call or return statement, the μ P must flush the pipeline and refill it with the instructions designated by the change-in-flow statement. These statements cause the μ P to assert the $\overline{\text{REFILL}}$ signal, signifying that the instruction pipeline is reloading.

The $\overline{\text{STATUS}}$ signal lets you externally trace the progress of the execution unit and certain exceptions as they occur. The external trace circuit ascertains existing conditions by counting the number of clock cycles when a $\overline{\text{STATUS}}$ signal occurs (Table 1).

Cache tracing is hit or miss

Typically, external circuitry is incapable of tracing access operations to a location in the internal cache memory of a μ P. However, the MC68030 provides an external address reference to locations in its on-chip cache memory. Because the 68030 implements a write-through policy, you can build a circuit that traces all write operations. If a cache hit occurs on a write cycle, the μ P will update both the external device and the location in the on-chip data cache referenced by the lower byte of the address word. If a cache miss occurs

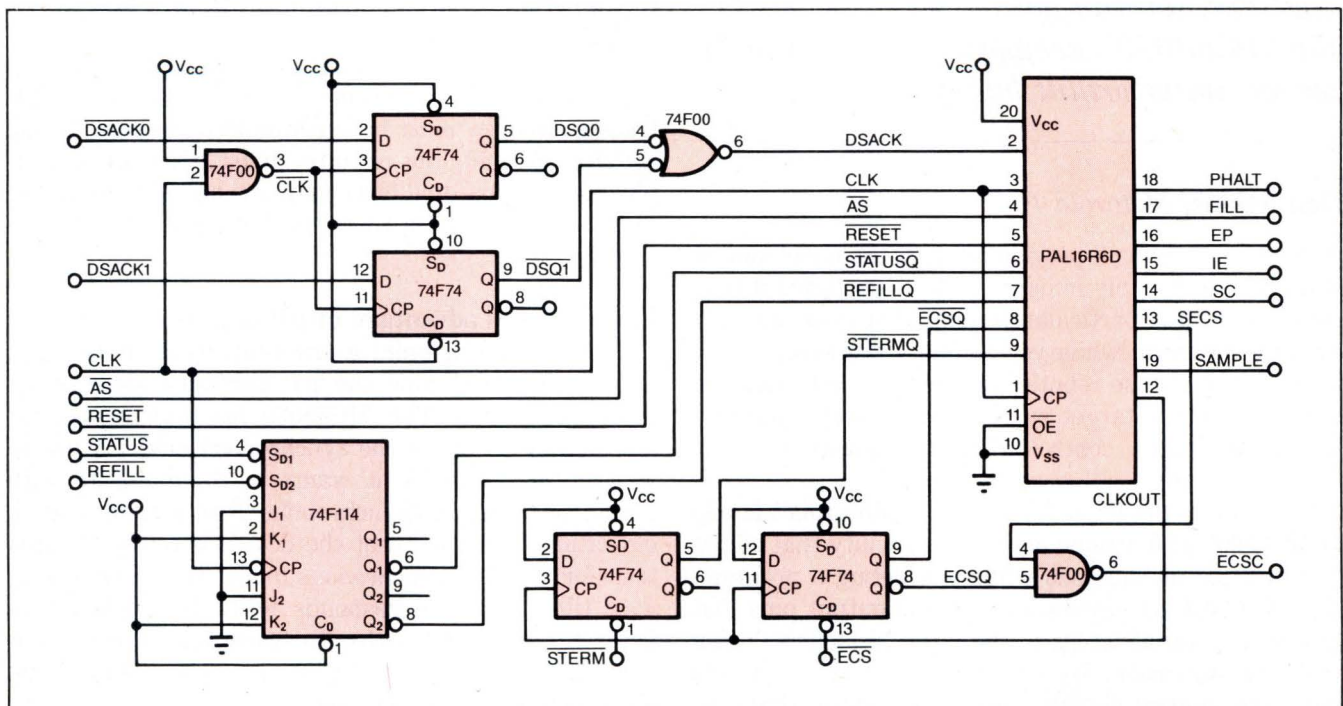


Fig 1—Developing the qualifying signals to externally trace program execution within a 68030 μ P and display it on a logic analyzer requires an interface circuit. Fortunately, you can build such a circuit with only a few ICs.

THIS DEVICE GENERATES A SAMPLING SIGNAL FOR TRACING PROCESSOR ACTIVITY ON AN INSTRUCTION LEVEL BASIS FOR THE MC68030. IN THE PIN DEFINITIONS AND EQUATIONS LISTED BELOW THE FOLLOWING SYMBOLS ARE USED:

SYMBOL DEFINITION
! LOGICAL NOT
LOGICAL OR
& LOGICAL AND

IN ADDITION, THE 'D' EXTENSION ON SIGNAL NAMES REFERS TO THE 'D' INPUT OF THE INTERNAL PAL FLIP-FLOP.

```

/** INPUTS */
PIN 1  = CLK      ; SAME AS PIN 3 CLK
PIN 2  = DSACK    ; DATA STROBE ACKNOWLEDGE
PIN 3  = CLK      ; MPU CLOCK SIGNAL
PIN 4  = IAS      ; ADDRESS STROBE
PIN 5  = IRESET   ; SYSTEM RESET SIGNAL
PIN 6  = ISTATUSQ ; LATCHED STATUS SIGNAL
PIN 7  = IREFILLQ ; LATCHED REFILL SIGNAL
PIN 8  = IECSQ    ; LATCHED ECS SIGNAL
PIN 9  = ISTERMQ  ; LATCHED STERM SIGNAL

```

```

/** OUTPUTS */
PIN 19 = SAMPLE   ; SAMPLE SIGNAL
PIN 18 = PHALT    ; PROCESSOR HALTED
PIN 17 = FILL     ; REFILL RECEIVED
PIN 16 = EP       ; EXCEPTION PENDING
PIN 15 = IE       ; INSTRUCTION EXECUTED
PIN 14 = SC       ; STATUS COMPLETE
PIN 13 = SECS     ; SAMPLED ECS SIGNAL
PIN 12 = CLKOUT   ; DELAYED CLK SIGNAL

```

(a)

```

/** INTERMEDIATE EQUATIONS */
STATE = PHALT SC EP IE
S0 = !PHALT & !SC & !EP & !IE; 0 = 0 0 0 0
S1 = !PHALT & !SC & !EP & IE; 1 = 0 0 0 1
S2 = !PHALT & !SC & EP & IE; 2 = 0 0 1 1
S3 = !PHALT & !SC & EP & !IE; 3 = 0 0 1 0
S4 = !PHALT & SC & EP & IE; 4 = 1 1 1 1
S5 = !PHALT & SC & !EP & IE; 5 = 0 1 0 1
S6 = !PHALT & SC & EP & !IE; 6 = 0 1 1 1
S7 = !PHALT & SC & EP & !IE; 7 = 0 1 1 0

```

```

/** LOGIC EQUATIONS */
!SAMPLE = !SC & !AS & !SECS #
          !SC & !DSACK & !STERMQ & !SECS #
          !SC & AS & !DSACK & !STERMQ & SECS;

```

```

!PHALT.D = !STATUSQ # !EP # IE # RESET;

```

```

!SC.D = RESET #
        S0 # S1 & STATUSQ #
        S2 & STATUSQ #
        S4 & !STATUSQ #
        SC & !PHALT;

```

```

!EP.D = RESET #
        S0 # S1 & !STATUSQ #
        S4 & !STATUSQ #
        SC & !PHALT;

```

```

!IE.D = RESET #
        S0 & !STATUSQ #
        S2 & STATUSQ #
        S3 & !STATUSQ #
        SC & !STATUS;

```

```

!SECS.D = !ECSQ;

```

```

!CLKOUT = !CLK;

```

```

!FILL.D = IREFILLQ & SAMPLE #
          !FILL & IREFILLQ #
          RESET;

```

(b)

on a write cycle, only the external device will be updated and no data will be entered into the data cache memory. If a hit occurs on a read cycle, the lower byte of the address word will provide an address reference to the location in the on-chip cache memory.

When the MC68030 wishes to transfer data to an external location, it places that particular address on the address bus and drives the ECS (External Cycle Start) signal low on the rising edge of the clock signal. If a hit occurs in one of the two internal cache memories or in the cache holding register, the μ P will abort the external-transfer cycle. During the transfer operation, the least significant byte on the address bus serves as the reference for where the hit occurred in the on-chip cache memory.

If no hit occurs, the μ P will assert the $\overline{AS}$ (Address Strobe) signal to validate the address before the next rising edge of the clock signal. The on-chip MMU (memory management unit) translates its virtual address to an external physical address. The MMU does not use the least significant byte (A_0 through A_7) in the translation process.

Fig 1 shows the hardware required to decode the signals from the 68030 so that you can debug a real-time program on a logic analyzer. All of the nine input signals, $\overline{DSACK0}$, $\overline{DSACK1}$, CLK, $\overline{AS}$, $\overline{RESET}$, $\overline{STATUS}$, $\overline{REFILL}$, $\overline{STERM}$, and $\overline{ECS}$, attach to the 68030 μ P in the system under development. The interface generates six output signals, PHALT, FILL, EP, IE, SAMPLE, and $\overline{ECS}$, to aid in capturing and analyzing real-time data. (A seventh output signal, SC, or Status Complete, is only used to generate state diagrams.) The logic analyzer connects to these signals to qualify information on the address and data bus. The system's CLK signal externally clocks the logic analyzer for synchronization. The logic analyzer captures the data on the falling edge of the CLK signal when the SAMPLE signal is high.

A PLD does most of the tracing

It is the programmable logic device (in this case the PAL16R6D) that generates the qualifying signals for logic analysis. Fig 2a defines the signals available on the pins of the PAL, and Fig 2b lists the PAL logic equations. Fig 3 is a state-machine diagram for the PAL.

Any one of the following five conditions will cause the interface hardware to assert the SAMPLE signal.

1. The beginning of an external bus cycle, indicated by the assertion of the $\overline{ECS}$ command.

Fig 2—The PAL16R6D logic device uses these defined input and output signals for its internal logic functions (a). You can use the accompanying logic equations (b) to program the PAL for the trace-interface circuit.

The trace-interface circuit generates a sampling signal to qualify the events and eliminate extraneous clock cycles.

2. A hit in the internal cache or cache holding register, indicated by the assertion of the $\overline{\text{ESC}}$ signal and without the $\overline{\text{AS}}$ command being asserted before the next clock cycle.

3. The encountering of an instruction boundary by the μP 's microsequencer.

4. The processing of an exception by the μP .

5. The occurrence of a double bus-fault, which automatically halts the μP .

The $\overline{\text{PHALT}}$ signal indicates a double fault on the system bus and signifies that the μP requires a reset signal to continue operation. The interface asserts the $\overline{\text{PHALT}}$ signal when the μP sets the $\overline{\text{STATUS}}$ signal for more than three cycles. The $\overline{\text{PHALT}}$ signal generates a SAMPLE signal.

The FILL signal indicates a break in the sequential execution of the program instructions. Actually, the FILL signal is a latched $\overline{\text{REFILL}}$ signal from the μP . It remains latched until the assertion of the next SAMPLE signal. The assertion of the FILL signal does not generate a SAMPLE signal.

The EP (Exception Pending) signal indicates that the MC68030 is beginning exception processing for either a reset, bus error, address error, spurious inter-

rupt, autovectored interrupt, F-line instruction, MMU address translation cache miss, or external trace. The EP signal asserts itself if the $\overline{\text{STATUS}}$ signal negates after two or three clock cycles. The assertion of EP generates a SAMPLE signal.

The IE (Instruction Executed) signal indicates the completion of an instruction by the μP 's execution unit. The μP asserts the $\overline{\text{STATUS}}$ signal for one clock period every time the execution unit finishes executing an instruction. The trace-interface circuitry detects this condition and issues the IE signal. The assertion of the IE signal also generates a SAMPLE signal.

The trace-interface circuit uses the $\overline{\text{SECS}}$ (Sampled $\overline{\text{ECS}}$) signal generated by the PAL and the latched $\overline{\text{ECS}}$ ($\overline{\text{ECSQ}}$) signal to derive the $\overline{\text{ECSC}}$ (External Cycle Start Condition) signal. The $\overline{\text{ECSC}}$ command, in conjunction with the $\overline{\text{AS}}$ signal, determines if the address bus and data bus are valid for the current SAMPLE signal. **Table 2** lists which bits are valid for the possible combinations of conditions for the $\overline{\text{AS}}$ and $\overline{\text{ECSC}}$ signals. The assertion of the $\overline{\text{ECSC}}$ signal does not generate a SAMPLE signal.

Although logic analysis is possible without a trace-interface circuit, it is difficult because you have to weed

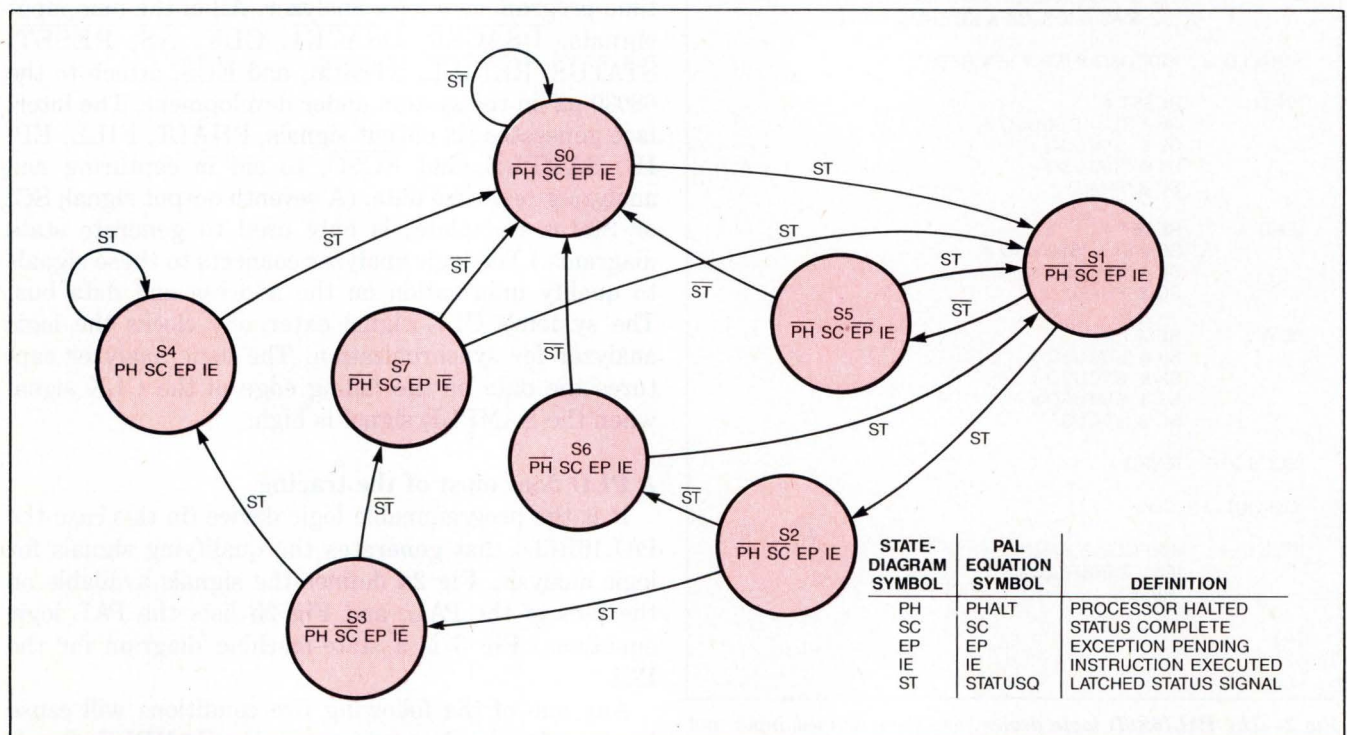


Fig 3—This state-machine diagram elucidates the logic equations of Fig 2b. The state of the latched STATUS signal (ST) determines which state transition occurs on each rising clock edge.

If a hit occurs on a read cycle, the lower byte of the address word will provide an address reference to the location in the on-chip cache memory.

TABLE 2—EFFECTS OF $\overline{AS}$ and $\overline{ECSC}$ SIGNAL ASSERTION

$\overline{AS}$	$\overline{ECSC}$	EFFECT
0	0	BOTH ADDRESS AND DATA BUS ARE VALID
0	1	BOTH ADDRESS AND DATA BUS ARE VALID
1	0	ADDRESS BITS (A_7-A_0) ARE VALID ADDRESS BITS ($A_{31}-A_8$) ARE INVALID DATA BUS IS INVALID
1	1	BOTH ADDRESS AND DATA BUS ARE INVALID

through so much data. The trace-interface circuit filters out the "noninteresting" clock cycles so that the logic analyzer displays only the cycles important for debugging. Listing 1, a logic-state listing, uses the trace-interface circuit to generate a display for a logic analyzer. In this example, 25 samples display a function that copies a source string to a destination string until the function encounters either a null character or a

hexadecimal 0. Without a trace-interface circuit, 99 samples show up when the logic analyzer takes a sample on every falling edge of the system clock. Because the extra 74 cycles occur when there is no external bus activity or when no instruction-execution information is provided by the $\overline{STATUS}$ and $\overline{REFILL}$ signals, they are not relevant to program debugging. **EDN**

Author's biography

Don Atkins is an applications design engineer for Motorola's high-end- μP semiconductor-products sector (Austin, TX). His duties include application support for the M68000 and M88000 μPs . He received a BSEE degree in 1982 from the University of Illinois. In his spare time, Don enjoys boating and waterskiing with his wife and two children.

Article Interest Quotient (Circle One)
High 494 Medium 495 Low 496

LISTING 1—TRACE-INTERFACE-CIRCUIT LOGIC-STATE PROGRAM

Index	Address	Data	$\overline{AS}$	PHALT	FILL	EP	IE	$\overline{ECSC}$	Comments
									Burst Read Prefetch
0	80000030	206F0004	0	0	0	0	0	1	move.l \$4(sp),a0
1	80000030	226F0008	0	0	0	0	0	1	move.l \$8(sp),a1
2	80000030	10D966FC	0	0	0	0	0	1	move.b (a1)+, (a0)+ bne.b \$-4
3	80000030	206F0004	0	0	0	0	0	1	move.l \$4(sp),a0
4	800020F8	FFFFFFFF	1	0	0	0	1	1	Instruction Executed
5	800020F8	80001100	0	0	0	0	0	1	Memory Write
6	800020F8	FFFFFFFF	1	0	0	0	1	1	Instruction Executed
7	80001100	30313233	0	0	0	0	0	1	Burst Operand Read
8	80001100	34353600	0	0	0	0	0	1	Burst Operand Read
9	80001100	00000000	0	0	0	0	0	1	Burst Operand Read
10	80001100	00000000	0	0	0	0	1	1	Instruction Executed
11	00003200	00000000	0	0	0	0	0	1	Byte Read
12	00003201	00000000	0	0	0	0	0	1	Byte Read
13	00003202	00000000	0	0	0	0	0	1	Byte Read
14	00003203	00FFFFFF	0	0	0	0	0	1	Byte Read
15	80001100	00000000	0	0	0	0	1	1	Instruction Executed
16	80001100	FFFFFFFF	1	0	0	0	1	1	Instruction Executed
17	80000040	FFFFFFFF	1	0	0	0	1	1	Instruction Executed
18	80000040	20084E75	0	0	0	0	0	1	move.l a0,d0
19	80000040	FFFFFFFF	0	0	0	0	0	1	rts
20	80000040	FFFFFFFF	0	0	0	0	0	1	-----
21	80000040	FFFFFFFF	0	0	0	0	0	1	-----
22	80000044	FFFFFFFF	1	0	0	0	1	1	Instruction Executed
23	80000044	FFFFFFFF	1	0	0	0	1	1	Instruction Executed
24	80000044	FFFFFFFF	1	0	0	0	1	1	Instruction Executed